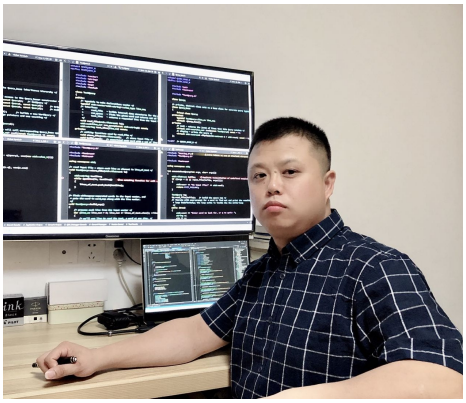
Frederique Hsu



Professional C++ developer and practised testing engineer with extensive experience of 18+ years.

</> Programming Skills

C++11/14/17/20	8+yrs
C, LabWindows/C	6+yrs
Embedded real-time C++	5+yrs
Python3	4yrs
C#	2.5yrs
Qt C++ Widget & QML	5yrs
Linux system/network prog.	6yrs
GoogleTest, Boost, Catch2, CppUTest, Unity	5yrs

Profile

Working from: Jul. 2006

Marriage: Single

Minhang district, Shanghai, China

+86 132 6283 5081

frederique.hsu@outlook.com

https://github.com/Frederick-Hsu

Languages & Comm.

English	Quite proficient, can talk as the 1st language on daily work.
Chinese	Native

Education background

Mar. 2016 - Mar. 2023 **Master, Computer science & technology** — Shanghai Jiao Tong University

- Thesis: Research on the technique of pulmonary bronchus segmentation in the CT-scanning images
- Model and implementation code: Airway3DSegmentNet (already opened in GitHub)

Sep. 2002 - Jul. 2006 **Bachelor, Mechatronics engineering** — Jimei University

Experience

Jul. 2023 - May. 2024 **C++ dev. expert (CAD geometry constraint solver engine)** — PoissonSoft Shanghai

- Participated in the development task about the constraint solver engine of domestic 3D CAD industrial design software, which has 2 core engines: Geometry Modeling, Geometric Constraint Solving. The GCS engine is used to efficiently solve the large scale of nonlinear equations, have the constraint conditions satisfied among various geometry elements, when engineer draws/designs with CAD software.
  - I had helped to design a hierarchical, message-passing based, and easy-to-horizontal-scale dynamically software architecture, according to the amount of connected graph components. This engine is programmed in C++17, consisted of such hierarchy layers as below:

1	User API layer
2	Geometry element(a.k.a. vertex) - constraint relationship(a.k.a. edge) graph model layer
3	Geometric decomposition layer according to connected component
4	Layer of converting constraint graph to nonlinear equations
5	Layer of jacobianizing nonlinear equations to solve, based on Newton-Raphson iteration method

  - Comparing with the benchmark Siemens D-Cubed engine in CAD industry, this engine has offered as rich features as 175+ interfaces, plus the toolsets to visualize the solving process and assess the effect. In the actual testing, when fully solving such a big nonlinear equations with 2M geometry elements and 30K constraints, it just costs less than 5 sec, 87% of D-Cubed performance. We have created the top performance index of domestic CAD GCS engine.

- Independently developed a suite of Sketcher visualization tool, by virtue of Qt Widget framework, and the CAD.3DViewer lite, they have integrated in the PCSC Engine, to validate the functionalities of engine inside our team, meanwhile test the actual solving effect and performance.
  - Additionally, took charge of the automatic CI building and Regression Test Pipeline, push the agile development with the optimized workflow: GitLab + Jira + ReviewBoard + CTest/CPack/CDash for all team players.

- Applied a patent: A kind of parallel accelerating method, applying the OpenGL hybrid heterogeneous computing on geometry constraint solving system.
  - This patent takes good use of OpenCL parallel heterogeneous computing framework for the 1st time, utilizing such computing resources as GPU. FPGA and DSP to accelerate the super-large-scale Jacobian matrix decomposition for geometry constraint solving system.
  - Applying this patent had dramatically improved the performance, decreasing the complexity of matrix multiplication from Strassen algorithm $O(n^{2.48})$ down to current $O(n^2 + n \log_2 n)$.

Mar. 2022 - Feb. 2023 **C++ developer (Autonomous driving firmware system)** — BMW (Shanghai) Autonomous Driving Lab.

- With my team to develop the Road Model module of L2 autonomous driving for BMW 3series, i.e. the experiment car gathering the driving trajectory data over such sensors as Lidar, mmRadar, Camera array, etc., then parse and fusion out the virtual lanes, relative to the physical lanes. They were submitted to the decision system, finally guide the EgoVehicle to drive along the virtual lanes. Visualize the trajectories, virtual lanes and video stream on the ROS RViz emulator, to validate and test the Road Model.
  - We have implemented the Road Model using C++ and Python on Ubuntu Linux platform, auto-built the software with Bazel build system, wrote many test scenarios in GoogleTest framework, orchestrated the Docker container cluster to make the full-scenarios testing for each feature.
  - After these test items, the firmware was programmed into domain controller, and executed the HIL, SIL testing, finally driver/vehicle-in-loop experiment.

- Actually I had participated or took responsible of these sub-modules of Road Model module as below:

1	RoadTrackerModule	2	TrajectoryTrackerModule	3	ClusteringModule
4	CentralFusionModule	5	LaneFinderModule	6	LaneTrackerModule

These sub-modules were executed like a pipeline, it is exactly the software architecture we've designed.

- RoadTrackerModule computes out the curved shape and boundary of road.
- TrajectoryTrackerModule calcuates the movement trajectory of each vehicle by using the Kalmann Filter algorithm, according to the position, direction and status of surrounding vehicles, which are relative to EgoVehicle, at every time slot. What's more, save these trajectories and update them.
- ClusteringModule adopts the Kruskal/Prim algorithms to classify the trajectories and aggregate into Left/Ego/Right 3 bunches.
- CenterFusionModule is responsible to fusion out the Ego lane, which will be used to guide the EgoVehicle.
- LaneFinderModule and LanTrackerModule then extracts the stable lanes from the trajectory bunches, and track/update the lanes in real-time.

- Implement the Lane Fusion & Path Guidance feature based on Tencent SD/HD map, splice the lane segments on the map together to form the legal driving path via the shortest path algorithm, then guide the EgoVehicle to drive along this path, including ramp up/down.

- Fix the software issues of Road Model module, write some sufficient test cases, maintain the Docker container auto-testing pipeline in data center, and take part in the prototype vehicle AD testing in Deqing testing ground and Shandong Highway to validate the software functionalities.

Jun. 2020 - Mar. 2022 **Senior C++ development engineer (Surgical robot controlling software)** MicroPort MedBot Surgical Robot

- Developed the 1st domestic laparoscopic surgical robot: Toumai. I took part in the development of 4 core modules as below: This system adopts the PLC ST(another kind of OOP language, but it features the hardware real-time as PLC) and C++ hybrid programming, based on Beckhoff TwinCAT platform, we've achieved the real-time manipulation from the master 2 arms on the surgeon cart to the slave 4 arms on the patient cart.

LogicControl	MotionControl	RobotKinematics	ForceControl

- Besides the development, we have collaborated with the surgeons from Shanghai Changhai hospital and SAHZU hospital to carry out the pancreatectomy, radical prostatectomy and hysteromyomectomy experiments on pigs and corpses, in order to validate its clinical applicability.
- Toumai finally passed the model certificate of NMPA on Jan. 27, 2022, as the sole domestic surgical robot.

- Developed the software system of Toumai surgical robot patient cart independently, I had adopted Qt QML to render the fancy and surgeon- friendly HMI, designed a well-organized architecture with 4-hierarchical layers: ControlUI—HMI logic—ADS Comm.—MotionCtrl, plus one independent Error Management module.
- I also proposed/implemented the innovative real-time mapping mechanism between State-Machine on driver layer and Mirror-State-Machine on UI application layer. This state-of-the-art software architecture realized the real-time response/control effect on both hardware and software.
- I used Qt QML/C++/Windows Driver hybrid programming to develop the software system.

Aug. 2016 - May 2020 **Software engineer** Shanghai Amphenol Airwave

- During the period I worked in SAA, I have accomplished these important projects as below:

1	Developed a comprehensive software platform and management suite in C# and C++: AUPS (Augmented Universal Platform & Sequenzer) It builds an integrated platform to support general-purpose automatic function testing, manages and executes the testing flow for communication electronic product. It also offers engineers the 2nd development for customization capability (This project had opened in GitHub.) ▸ AUPS utilizes the sequence.xml test script to save and organize test items, manages their automation flow, provides the graphical editor to allow editing items and parameters by engineers themself. ▸ Design an uniform UI to monitor the automatic testing process. ▸ AUPS achieved up to 16 sessions running in parallel, multiple instruments can be connected together over the ethernet, to compose the auto-testing production line, AUPS facilitated to improve automation dramatically.
2	Wrote out a set of class library to access various instruments, as long as the instruments support VISA standard, which covers a wide range of instrument models from Keysight, R&S and Tektronix etc.
3	Designed the NFC data link equipment for programming firmware image to iPhone 8 & X smart phones in the contactless way. We had applied a patent for this equipment: One kind of NFC data transferring device with ultra-large capability.
4	Based on the AUPS software, I developed a suite of integrated utility, to help GM's Chevrolet, Cadillac SUV to carry out the automatic testing about V2X vehicle communication, and traveled to Detroit GM tech. center to guide the testing and development.

Feb. 2012 - Jun. 2016 **Senior software dev./testing engineer** Hella Shanghai Electronics

- Developed the production automatic testing system and equipment, based on NI LabWindows/CVI(pure ANSI C), including testing software development, hardware controlling system design and integration . I had accomplished the following important testing system projects as below:
  - Peugeot-Citroën RKE remote key testing system
  - Volkswagen FKS12 & Kessy passive entry passive start product testing equipment
  - BMW & MINI remote key (both RKE and PEPS PKE) automatic testing system
  - The Insulation Monitoring Device testing system development for Mercedes-Benz & BYD jointed DENZA battery-electric-vehicle

- Developed a compact integrated test instruments DCU, in order to replace the expensive NI PXI-card instruments. Based on Renesas V850 uPD70F3376 MCU, I wrote only by myself the DCU_MCTBox_Firmware, DCU_MCTBox_API and DCU_MCTBox_Diagnoser utility software. This DCU provides such functions as SwitchMatrix, DIO, DigitMeter, DigitVoltSource, CAN/LIN analyzer etc., which can meet the basic requirement of automotive electronics product.
  - This project had fully opened in GitHub, expect that developing the open-source, cheap but standard-shared testing instruments with open community.

Oct. 2009 - Feb. 2012	**Test software engineer**	Shanghai Foxconn
Jul. 2006 - Aug. 2009	**LCD TV firmware engineer**	TPV AOC (Fuzhou)

Projects

PoissonSoft Constraint Solver Engine — PoissonSoft

Proj. description and Responsibility:

- Developed the domestic CAD constraint solver engine, compared with the benchmark Siemens D-Cubed.
- Design and implement some local APIs as below and cloud interfaces `PSCSCloudAPI` in C++17, and the replaying feature of solving log scripts.
`PSCSApiSolver2DEngine::isLicenseValid`, `PSCSApiSolver2D::measureDimension`, `incrementalEvaluate`, `dragging`, `overConstraintAnalyze` etc.
- Coding: 55%, Testing and performance parsing: 15%, Write Wiki document and design solution: 20%, Fixing issues: 10%
- Develop the visual utility Sketcher and PSCAD.3DViewer (with Qt widgets, Qt3DRender) and framework.cad
- Studying on contrast the innovative solving algorithms of open-source OpenCascade, compare the difference between OCCT, PSCS Engine and D-Cubed in terms of implementation and solving.
- Study how to vectorize the large-scale Jacobian matrix, combining with OpenCL heterogeneous parallel computing, deploy it onto the GPU to improve parallelism.
- Maintain the CMake scripts, automate the Build/Test/Pack/Deploy workflow, have PSCS engine worked on Windows/Linux/macOS cross-platform and HuaweiCloud, meanwhile compatible with x86_64 CPU and Huawei Kunpeng ARM CPU.

Tech. stack I used:

Graph model connected component
nlohmann-json serialization & deserialization
Eigen MatrixLib Qt3DRender
OpenCascade solving
OpenCL heterogeneous parallel computing
GoogleTest Vistor Pattern
CMake

Some important algorithms:

Newton-Raphson iteration method
Trust-Region method

Autonomous Driving Perception/Road Model module — BMW

Proj. description and Responsibility:

- Develop the Road Model module of Perception system of L2 Autonomous Driving for BMW 3series, to meet the requirement of China market.
- Parse in-depth the trajectories of surrounding vehicles, which were sensored by Lidar, mmRadar and camera array. Utilizing Kalmann Filter algo. to fit/calibrate/identify those trajectories, then aggregate them into Left/Ego/Right lane bunches, finally fusion out the virtual lanes, relative to the physical lanes.
- Playback the fitted trajectories and the real-time video stream on ROS RViz emulator, check whether the fitted trajectories are approximating the genius vehicle's trajectories, especially on the curved road or lane-changing, in some extreme scenarios of road forks and merging, to filter out the distorted trajectories.
- Coding: 40%, Testing(Unit test + Integrated test + HIL + SIL): 30%, Design the solution: 15%, Experiment on testing ground and highway: 15%
- Generate the random driving scenarios in Python script, then carry out the Fuzzy Test, log the test failures.
- Gather the video stream which are collected from experiment cars and BMW vehicles, stored in the data center, orchestrate the Docker containers to testing pipeline, make the extensive testing for each feature and full scenarios.
- Write the publisher/subscriber topics in the ROS environment, develop the robot nodes for EgoVehicle and surrounding vehicles, Simulate in the ROS RViz and visualize the real effect.

Tech. stack I used:

Google Bazel incremental build system
GoogleTest
ROS RViz emulator
Docker containers orchestrating

Some important algorithms:

Kalmann Filter algorithm
Kruskal, Prim algorithm to find the min splay tree, aggregate the trajectory bunches.